

Secure Coding for SQL Injection Prevention Using Generative AI

Young-Bok Cho*

*Professor, Dept. of Computer Education, Andong National University, Andong, Korea

[Abstract]

In this paper, Generative AI is a technology that creates various forms of content such as text, images, and music, and is being utilized across different fields. In the security sector, generative AI is poised to open up new possibilities in various areas including security vulnerability analysis, malware detection and analysis, and the creation and improvement of security policies. This paper presents a guide for identifying vulnerabilities and secure coding using ChatGPT for security vulnerability analysis and prediction, considering the application of generative AI in the security domain. While generative AI offers innovative possibilities in the security field, it is essential to continuously pursue research and development to ensure safe and effective utilization of generative AI through in-depth consideration of ethical and legal issues accompanying technological advancements.

▶ **Key words:** Generative AI, ChatGPT, Secure coding, SQL injection, Web vulnerability

[요 약]

생성형 인공지능은 텍스트, 이미지, 음악 등 다양한 형태의 콘텐츠를 생성하는 기술로, 다양한 분야에 활용되고 있다. 보안 분야에서도 생성형 인공지능을 활용한 디지털포렌식, 악성코드 분석, 취약점분석 등 다양한 분야에 새로운 가능성을 열 준비를 하고 있다. 본 논문에서 생성형 인공지능을 보안영역을 고려한 활용방법으로 보안 취약점 분석 및 예측을 위해 ChatGPT를 활용해 취약점 식별 및 시큐어 코딩이 가능한 가이드를 제시한다. 생성형 인공지능은 보안 분야에 혁신적인 가능성을 제시하지만, 기술적 발전과 함께 윤리적, 법적 문제에 대한 심도있는 고민을 통해 생성형 인공지능이 안전하고 효과적으로 활용될 수 있도록 지속적인 연구와 개발이 필요할 것이다.

▶ **주제어:** 생성형 인공지능, 챗GPT, 시큐어 코딩, SQL 인젝션, 웹 취약점

I. Introduction

대챗GPT(ChatGPT)는 OpenAI에서 개발한 대화형 인공지능 서비스로, 채팅(Chat)과 대형언어모형(GPT: Generative Pre-trained Transformer)의 합성어로 GPT는 대규모 자연어 데이터를 학습해 질문에 대한 답변을 생성할 수 있도록 설계된 인공지능 모델로, 현재 GPT-4까지 발전되었대[1]. 인간의 언어를 이해하고 생성하는 능력이 뛰어나, 다양한 분야에서 광범위하게 활용되고 있는데 ChatGPT는 특히 사용자와 자연스러운 대화를 나누고, 다양한 질문에 대한 답변을 제공하며, 복잡한 문제를 해결하는 데 많은 도움을 주고 있다[1,2]. 최근 들어, ChatGPT와 같은 생성형 인공지능 모델의 활용 범위가 급격히 확장되면서 교육, 헬스케어, 금융, 법률 등 여러 분야에서 ChatGPT는 정보 제공, 문제 해결, 사용자 경험을 향상하는 도구로 자리 잡고 있다. 교육 분야의 경우에는 학생들의 학습 도우미로, 헬스케어 분야에서는 환자 상담 도구로, 금융 분야에서는 고객 서비스 및 데이터 분석 도구로 활용되고 있다[3]. 이러한 활용 사례들은 ChatGPT의 유용성을 입증하는 동시에, 그 한계와 잠재적 위험성에 대한 논의도 함께 불러일으키고 있지만 ChatGPT의 광범위한 활용과 함께 생성형 인공지능 플랫폼의 문제점과 보안 취약성도 주목받고 있다. 생성형 인공지능 모델이 생성하는 콘텐츠는 종종 부정확하거나 편향된 정보를 포함할 수 있으며, 악의적인 사용자가 이를 악용할 가능성도 존재하는데 특히, 소프트웨어 개발 및 보안 분야에서 ChatGPT와 같은 생성형 인공지능의 사용은 신중하게 고려되어야 한다[4].

소프트웨어의 보안 취약점을 악용한 사이버 공격은 막대한 피해를 초래할 수 있으므로, 개발 단계에서부터 보안을 철저히 고려하는 시큐어 코딩(Secure Coding)은 더 이상 선택이 아닌 필수가 되었다. 시큐어 코딩은 프로그램 개발 과정에 발생할 수 있는 보안상 취약점을 최소화하고, 안전한 소프트웨어를 개발하기 위한 일련의 방법론과 기술로 단순히 코드 작성 단계에서의 보안 강화를 넘어서, 설계, 구현, 테스트, 배포 등 소프트웨어 개발의 전 과정에 걸쳐 보안을 고려하는 것을 포함한다[5,6]. 우리나라의 경우 한국인터넷진흥원(KISA)은 프로그램 개발자들의 안전한 개발을 지원하기 위해 '소프트웨어 개발 보안 가이드'를 발간하고, 개발자들이 보안 취약점을 예방할 수 있도록 다양한 교육과 자료를 제공하고 있다[7]. 본 논문에서는 ChatGPT를 활용해 시큐어 코딩을 적용하는 가능성의 실험을 통해 증명하고 향후 최종적으로는 생성형 인공지능

을 활용한 시큐어 코딩의 실질적인 적용 방안과 그 한계점에 대해 고찰한다.

본 논문의 구성은 2장에서 생성형 인공지능과 시큐어 코딩, 그리고 이 두 영역의 결합에 관한 주요 연구를 살펴보고 3장에서는 ChatGPT를 사용해 보안 취약점에 대해 해결책을 실습으로 증명하고 4장에서는 ChatGPT를 이용해 시큐어 코딩을 수행시 장단점을 평가한다. 마지막으로 5장에서는 결론으로 구성한다.

II. Preliminaries

생성형 인공지능, 특히 GPT-3와 같은 대형 언어 모델은 최근 몇 년간 다양한 연구 분야에서 중요한 혁신을 가져왔고, 이러한 모델들은 자연어 처리(NLP) 작업에서 놀라운 성과를 보여주었으며, 텍스트 생성, 번역, 요약, 질의응답 등 다양한 분야에서 활용되고 있다.

1. Generative AI

OpenAI의 GPT-3는 1,750억 개의 매개변수를 가진 대형 언어 모델로, 그 이전 버전인 GPT-2의 15억 개 매개변수에 비해 비약적으로 향상된 성능을 보여주고 있다. GPT-3는 다양한 자연어 처리 작업에서 우수한 성능을 보이며, 텍스트 완성, 텍스트 번역 및 요약 그리고 질문 응답 등의 작업에서 인간과 유사한 결과를 생성할 수 있다[8]. ChatGPT는 이러한 GPT-3의 성능을 기반으로 한 대화형 인공지능으로 사용자와 자연스러운 대화를 통해 정보를 제공하고 문제를 해결하는 데 활용된다. 초거대 인공지능 모델에서 매개변수 최적화는 모델의 성능을 좌우하게 되는데 매개변수가 많다고 무조건 성능이 좋은 것은 아니지만 일반적으로 양질의 매개변수가 많을수록 성능은 우수하다[2, 9]. 생성형 인공지능은 여러 분야에서 응용되고 있다. 예를 들어, 의료 분야에서는 환자 상담과 진단 지원, 금융 분야에서는 고객 서비스 자동화와 데이터 분석, 교육 분야에서는 맞춤형 학습 도우미 등으로 활용되고 있는데 특히, 교육 분야에서 활용은 학생들의 학습 경험을 개인화하고, 다양한 학습 스타일에 맞춘 콘텐츠를 제공함으로써 학습 효과를 높이는 데 기여하고 있다.

2. Secure Coding

시큐어 코딩은 프로그램 개발자들이 소프트웨어 개발 과정에서부터 보안(security)과 취약점(vulnerability)을 최소화하고 안전한 소프트웨어를 개발하기 위한 일련의

Table 1. Software Development Security Guide

Input Data Validation and Presentation(17)			Security Features (16)			Time and State(2)	Error Handling (3)	Code Defects (5)	Encapsulation (4)	API Misuse (2)
SQL Injection	Untrusted URL Redirection	Server Side Request Forgery	Allowing Critical Functionality without Proper Authentication	Use of Weak Encryption Algorithm	Sensitive Information Not Encrypted	Race Condition: Time of Check to Time of Use (TOCTOU)	Information Exposure Through Error Messages	Null Pointer Deference	Information Exposure Through Incorrect Session Handling	Security Decision Relying on DNS Lookup
code Injection	Improper XML External Entity Reference	Integer Overflow	Use of Insufficient Key Size	Allowing Weak Passwords	Hardcoded Sensitive Information	Unterminated Loop or Recursive Function	Lack of Error Handling	Improper Resource Cleanup	Debug Code Left in Production	use of vulnerable API
LDAP Injection	Improper Input Used for Security Decision	Memory Buffer Overflow	Sensitive Information in Comments	Use of Inadequate Random Values	Improper Digital Signature Verification		Improper Exception Handling	Use of Freed Resource	Private Array Returned from Public Method	
XML Injection	Cross Site Scripting	Dangerous File type Upload	Improper Permissions for Critical Resources	Use of One-Way Hash Function without Salt	Code Download Without Integrity Check			Use of Uninitialized Variable	Assignment of Public Data to Private Array	
HTTP Response Splitting	Dangerous File Type Upload	Path Manipulation and Resource Injection	Information Exposure through Cookies Stored on Disk	Lack of Brute Force Protection				Deserialization of Untrusted Data		
Format String Injection	Cross Site Request Forgery		Improper Certificate Validation	Use of Weak Cryptographic Algorithm						

방법론과 기술을 의미하는 것으로 코드 작성 단계의 보안 강화를 넘어 설계, 구현, 테스트, 배포 등 소프트웨어 개발의 전 과정에 걸쳐 보안을 고려하는 것을 포함한다. 시큐어 코딩의 필요성은 사이버 공격의 증가와 이에 따른 보안 사고의 심각성에서 비롯되고 소프트웨어 보안 취약점은 악의적인 공격자가 이를 악용하여 시스템에 침입하거나 데이터를 유출하는 등의 피해를 초래할 수 있다. 시큐어 코딩을 지원하기 위해 다양한 기준과 도구들이 개발되었는데 대표적으로, 한국인터넷진흥원(KISA)은 ‘소프트웨어 개발 보안 가이드’를 발간해 개발자들이 보안 취약점을 예방할 수 있도록 지원하고 있다[7]. 이 가이드는 [분석 및 설계] 단계에서 보안 강화를 통해 시큐어 코딩을 위한 방법을 제시하며 [구현] 단계에서는 “입력 데이터 검증, 보안 기능, 시간 및 상태 관리, 에러 처리, 코드 오류, 캡슐화, API 오용 등” 7가지 영역에서 보안 취약점 49개를 통제하고 위협(threat)을 최소화하는 방안을 제공한다. [표 1]은 KISA에서 제공하는 ‘소프트웨어 개발 보안 가이드’로 시

큐어 코딩에 고려될 내용을 나타낸 것이다. 일반적으로 소프트웨어를 구성하는 소스 코드의 보안 취약점을 점검하는 방식은 소스 코드 내에서 잠재적인 보안 취약점을 분석하는 정적 분석기술과 구동되고 있는 애플리케이션을 임의로 공격해 공격의 성공 여부를 확인하는 동적 분석기술로 구분된다. OWASP(Open Web Application Security Project)에서는 주요 웹 애플리케이션 보안 취약점을 정리한 ‘OWASP Top 10’을 발표하고 이를 기반으로 한 보안 코딩 가이드라인을 제공하고 있다. 이 가이드라인과 더불어 정적 분석 도구(Static Analysis Tools)와 동적 분석 도구(Dynamic Analysis Tools) 등 다양한 보안 도구들이 시큐어 코딩을 지원한다[10, 11].

3. The Integration of Generative AI and Secure Coding

생성형 인공지능은 시큐어 코딩을 지원하는 데 있어 다양한 가능성을 제시한다. 예를 들어, GPT-3와 같은 언어

모델은 소프트웨어 코드에서 잠재적인 보안 취약점을 식별하고, 이를 수정하기 위한 권장 사항을 제공할 수 있고, 개발자가 작성한 코드의 보안성을 자동으로 검토하고, 보안 강화를 위한 코드 리팩토링 방안을 제안하는 데 활용될 수 있다. 이러한 기능들은 개발자들이 보안 문제를 사전에 인지하고, 이를 예방하는 데 큰 도움을 줄 수 있다[10,13].

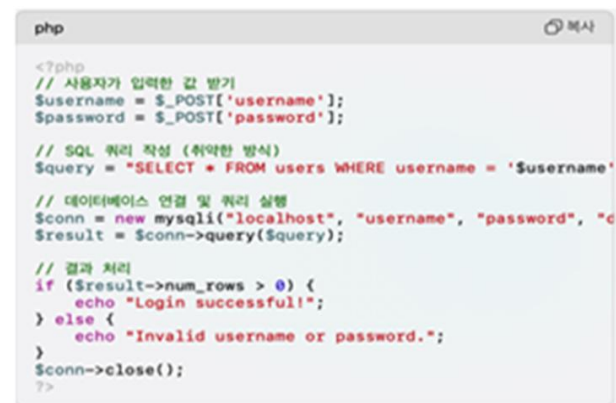
최근 연구들은 생성형 인공지능이 시큐어 코딩에 어떻게 이바지할 수 있는지를 다양한 관점에서 살펴볼 수 있는데 한 연구에서는 GPT-3를 활용해 코드의 보안 취약점을 자동으로 식별하고 수정하는 방법을 제안하였다. 연구 결과, GPT-3는 SQL 인젝션(SQL 인젝션), 크로스 사이트 스크립팅(XSS) 등의 주요 보안 취약점을 효과적으로 식별하고, 이를 수정하는 코드를 생성할 수 있음을 보였고[14]. 다른 연구에서는 생성형 인공지능을 활용한 자동 코드 검토 시스템을 개발해, 코드 검토 과정에서 발생할 수 있는 보안 취약점을 사전에 방지할 수 있음을 입증하였다[13, 14]. 생성형 인공지능은 시큐어 코딩 교육에서도 유용하게 활용될 수 있다. 예를 들어, 교육용 챗봇을 통해 학생들이 시큐어 코딩 원칙을 학습하고, 실제 코드 작성 과정에서 보안 취약점을 인지하고 수정하는 능력을 배양할 수 있고, 생성형 인공지능을 활용한 쌍방향 학습 도구는 학생들이 다양한 보안 시나리오를 체험하고, 이에 대응하는 능력을 키우는 데 도움을 줄 수 있다[11, 16].

III. The Proposed Scheme

시큐어 코딩을 위한 정적 분석(static analysis) 기술을 이용한 진단 결과에 많은 취약점이 도출되는 대표적인 영역으로는 '입력 데이터 검증 및 표현'이 있다. 개발 단계에서 입력값을 철저히 검증해 소프트웨어가 사용자의 입력을 처리하는 과정에서 의도치 않은 결과가 발생하지 않도록 해야 한다[5]. 또한 입력 데이터의 검증 및 표현과 관련해 SQL 인젝션, 크로스 사이트 스크립팅(XSS), 운영 체제 명령어 삽입, 그리고 위험한 형식의 파일 업로드와 같은 문제들이 빈번히 발생하게 되는데 이러한 문제 중 SQL 인젝션 취약점을 다루기 위해 ChatGPT를 활용할 수 있는 방법은 다음과 같다.

1. Mitigating SQL Injection Vulnerabilities Using Generative AI

제안된 SQL 인젝션은 애플리케이션에서 사용자의 입력값을 악용해 서버의 데이터베이스를 공격하는 기술로 입력값에 대한 유효성 검증이 부족할 때 발생하고 의도하지 않은 질의문이 DB에 실행되어 데이터를 무단으로 열람하거나 조작할 수 있게 만드는 보안 취약점을 가리킨다[6]. ChatGPT에 'SQL 인젝션 문제가 있는 코드를 보여줘'라고 입력하면, 취약한 코드 예시와 이를 해결하기 위한 안전한 코드 방안을 함께 제공받을 수 있다. [그림 1]은 PHP로 구현된 로그인 기능에서 발생할 수 있는 SQL 인젝션 예를 나타낸 것으로 사용자가 \$username에 'admin'--'를 입력하는 경우 Oracle, MSSQL, MariaDB에서는 '주석 처리'를 의미하는 '--' 문자로 인해 \$password가 올바른 값이 아니더라도 로그인이 가능해진다. 이러한 방식은 SQL 질의문을 변조해 인증 절차를 우회할 수 있는 취약점이 발생할 수 있다. 이와 같은 SQL 인젝션 취약점 대응을 통해 시큐어 코딩을 구현하는 방법을 ChatGPT에서 요청할 수 있다.



```

php
<?php
// 사용자 입력값 받기
$username = $_POST['username'];
$password = $_POST['password'];

// SQL 쿼리 작성 (취약한 방식)
$query = "SELECT * FROM users WHERE username = '$username'";

// 데이터베이스 연결 및 쿼리 실행
$conn = new mysqli("localhost", "username", "password", "c");
$result = $conn->query($query);

// 결과 처리
if ($result->num_rows > 0) {
    echo "Login successful!";
} else {
    echo "Invalid username or password.";
}
$conn->close();
?>

```

Fig. 1. An example of SQL injection that can occur in the login function written by ChatGPT

본 연구에서 자바로 구현된 SQL 인젝션 사례와 해결 방안을 질의한 결과 SQL 인젝션을 방지하기 위해 Statement보다 매개변수 바인딩을 지원하는 PreparedStatement 사용을 권장했다. 따라서 ChatGPT를 이용해 "자바 기반의 SQL 인젝션 예제를 알려줘"라고 요청한 결과[그림 2]와 같은 코드를 획득할 수 있었다.

```

java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class SQLInjectionExample {

    public static void main(String[] args) {
        // 사용자가 입력한 값 (취약점 발생 가능)
        String username = "admin";
        String password = "' OR '1' = '1";

        try {
            // 데이터베이스 연결
            Connection connection = DriverManager.getConne
            // 취약한 SQL 쿼리 (사용자의 입력을 직접 삽입)
            String query = "SELECT * FROM users WHERE user
            // 쿼리 실행
            Statement stmt = connection.createStatement();
            ResultSet rs = stmt.executeQuery(query);

            // 결과 처리
            if (rs.next()) {
                System.out.println("Login successful! User
            } else {
                System.out.println("Invalid username or pa
            }

            // 연결 종료
            connection.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
    
```

Fig. 2. Example of SQL injection vulnerability code when using the Statement provided by ChatGPT

[그림 2] 코드에서 사용자 입력이 직접 SQL 질의문에 query="SELECT * FROM users WHERE username = ' + username + ' AND password = ' + password + '";"가 같이 삽입되어 있어 이를 통해 공격자는 사용자 이름 또는 비밀번호 필드에 악의적인 SQL 코드를 삽입할 수 있는 취약점이 존재한다. 예를 들어 사용자 이름 필드에 'admin'--'값을 입력하면 위 코드를 SQL 질의문이 [그림 3]과 같이 생성된다.

```

sql
SELECT * FROM users WHERE username = 'admin' AND password = '' OR '1' = '1';
    
```

Fig. 3. Vulnerabilities Caused by User Input During SQL Query Generation

[그림 3]의 질의문은 '--' 이후의 모든 텍스트를 주석 처리하므로 실제로는 username = 'admin'으로 인해 admin 사용자의 패스워드 검증 없이 로그인에 성공할 수 있게 되고 이를 통해 악의적인 사용자는 로그인 없이 관리자 권한으로 시스템을 조작할 수 있게 된다. 따라서 SQL 질의문 생성 시 사용자 입력을 직접 삽입하는 것은 취약점으로 이어질 수 있으므로 시큐어 코딩을 통한 SQL 질의문 생성 방법을 사용하면 취약점을 방지할 수 있는데 PreparedStatement에서는 물음표(?) 자리에 사용자가 입력한 값이 그대로 들어가지만, 데이터베이스 엔진은 이를

질의문의 일부가 아닌 데이터로만 인식하게 된다. 즉, 특수문자(' OR '1' = '1 등)가 그대로 문자열로 처리되어 SQL 구문의 논리적인 실행에는 영향을 미치지 못하게 되기 때문에 특수문자의 자동 파싱을 통해 SQL 인젝션에 대응할 수 있는 PreparedStatement의 사용을 권장하고 있다. ChatGPT를 통해 안전한 시큐어 코딩 방법을 요청한 결과[그림 2]의 코드 중 사용자 입력값을 안전하게 처리하도록 PreparedStatement를 사용해 String query = "SELECT * FROM users WHERE username = ? AND password = ?";와 같이 안전한 SQL 질의문 작성을 제시해 주었다. 그러나 소프트웨어의 목적에 따라 어쩔 수 없이 Statement를 사용해야 할 때에는 이에 대한 추가적인 대응 방안을 마련할 필요는 있을 것이다.

조건절이 매번 달라지는 동적 SQL에서는 성능과 코드 작성 측면에서 PreparedStatement보다 Statement를 사용하는 것이 더 효율적이므로 ChatGPT를 통해 "Statement를 사용하면서 SQL 인젝션을 탐지할 수 있게 해줘"라고 다시 질문한 결과 "Statement를 사용하면 SQL 인젝션 공격을 탐지하기 어렵다. 그러나 Statement 대신에 PreparedStatement를 SQL 인젝션 공격을 탐지할 수 있습니다"라는 답변과 함께 [그림 4]와 같은 코드를 얻을 수 있었다.

```

// SQL 인젝션 패턴 정의
private static final Pattern SQL_INJECTION_PATTERN = Pattern.compile(
    "(?i);--|\\bOR\\b|\\bAND\\b|\\bXOR\\b|\\bSELECT\\b|\\bINSERT\\b|\\bUPDATE\\b|\\bDELETE\\b|",
    Pattern.CASE_INSENSITIVE);

public static void main(String[] args) {
    // 사용자 입력 (검증 필요)
    String username = "admin";
    String password = "' OR '1' = '1";

    // 입력 검증
    if (isPotentiallyMalicious(username) || isPotentiallyMalicious(password)) {
        System.out.println("Potential SQL injection detected in input.");
        return;
    }

    try {
        // 데이터베이스 연결
        Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "root", "pass

        // SQL 쿼리 생성
        String query = "SELECT * FROM users WHERE username = " + username + " AND password = " + password +

        // 쿼리 실행
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(query);

        // 결과 처리
        if (rs.next()) {
            System.out.println("Login successful! User: " + rs.getString("username"));
        } else {
            System.out.println("Invalid username or password.");
        }

        // 연결 종료
        connection.close();
    }
}
    
```

Fig. 4. Example of adding SQL injection detection and logging to vulnerable code (using Statement) by ChatGPT

[그림 4] 코드에서 입력값이 질의문에 의심스러운 패턴을 포함하고 있는지 확인하기 위해 SQL 인젝션에서 가장

일반적으로 사용되는 패턴을 정규식으로 먼저 정의하고, 입력값을 isPotentiallyMalicious 메서드를 사용해 패턴 여부를 판별하는데 패턴이 발견되면 쿼리 실행을 중단하고 잠재적인 SQL 인젝션을 경고하게 되고, 만약 입력값이 정상으로 검증이 완료되면 질의문을 실행하게 된다. 이 접근 방법만으로 완전한 SQL 인젝션을 방지할 수는 없으나 사용자의 입력을 검사해 잠재적인 공격을 탐지하고 방지하는 데는 도움이 될 수 있어 PreparedStatement와 같은 안전한 질의문 작성 방법이 가장 효과적인 방지 방법임을 제시하였다. [그림 5]는 ChatGPT가 PreparedStatement를 사용해 안전한 질의문 코드를 제시한 결과로 SQL 질의문과 사용자 입력값이 명확히 분리되어, SQL 인젝션 공격을 효과적으로 방지할 수 있고, 입력값을 SQL 질의문의 일부로 처리하지 않기 때문에, 공격자의 악의적인 SQL 코드가 포함되더라도 질의문의 구조를 변경할 수 없게 된다.

2. SQL Injection Vulnerabilities in Generative AI

```
public static void main(String[] args) {
    // 사용자 입력값 (안전하게 처리됨)
    String username = "admin";
    String password = "securePassword"; // 사용자의 입력이지만 안전하게 처리됨

    try {
        // 데이터베이스 연결
        Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "root", "pass");

        // PreparedStatement를 사용하여 안전한 SQL 쿼리 작성
        String query = "SELECT * FROM users WHERE username = ? AND password = ?";

        // PreparedStatement로 쿼리 준비 및 사용자 입력값 바인딩
        PreparedStatement pstmt = connection.prepareStatement(query);
        pstmt.setString(1, username);
        pstmt.setString(2, password);

        // 쿼리 실행
        ResultSet rs = pstmt.executeQuery();

        // 결과 처리
        if (rs.next()) {
            System.out.println("Login successful! User: " + rs.getString("username"));
        } else {
            System.out.println("Invalid username or password.");
        }

        // 자원 정리
        rs.close();
        pstmt.close();
        connection.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Fig. 5. Safe query writing code (using PreparedStatement) by ChatGPT

실제 서버 환경에서 SQL 인젝션 취약점을 분석해 보기 위해 오픈 소스 기반의 웹사이트 제작 플랫폼인 '워드프레스(WordPress.org)'에서 발생한 SQL 인젝션 취약점의 소스 코드를 분석한다. WordPress에서 admin-ajax.php 파일은 Ajax 요청을 처리하는 데 사용되는 중요한 엔드포인트로 WordPress 백엔드에서 다양한 Ajax 요청을 처리하며 wp_ajax와 wp_ajax_nopriv로 구분되어 각각 인

```
// 인증되지 않은 사용자도 호출할 수 있도록 wp_ajax_nopriv_ 사용
add_action('wp_ajax_nopriv_my_custom_action', 'my_custom_action_handler');

function my_custom_action_handler() {
    // 인증되지 않은 사용자의 요청 처리

    // 보안 검증 없이 사용자가 입력한 값을 처리할 경우 취약점 발생 가능
    $data = $_POST['data']; // 사용자로부터 입력받은 값

    // 데이터 처리 (보안 취약 가능성 있음);
    echo 'Received data: ' . $data;

    wp_die(); // Ajax 요청 후 종료
}

(a) Before admin-ajax.php code to allow requests from unauthenticated users

// 인증되지 않은 사용자도 호출할 수 있는 Ajax 액션 추가
add_action('wp_ajax_nopriv_my_custom_action', 'my_custom_action_handler');

function my_custom_action_handler() {
    // CSRF 보호: 비일 nonce 확인
    if (!isset($_POST['nonce']) || !wp_verify_nonce($_POST['nonce'], 'my_custom_nonce')) {
        wp_send_json_error('Invalid request');
        wp_die();
    }

    // 입력값 검증
    $data = isset($_POST['data']) ? sanitize_text_field($_POST['data']) : '';

    if (empty($data)) {
        wp_send_json_error('No data provided');
        wp_die();
    }

    // 안전하게 데이터 처리
    echo 'Received sanitized data: ' . esc_html($data);

    wp_die(); // Ajax 요청 후 종료
}

(b) Secure code admin-ajax.php
```

Fig. 6. Before and after secure coding of admin-ajax.php in WordPress

증된 사용자와 인증되지 않은 사용자의 요청을 처리하게 된다. 인증되지 않은 사용자의 요청을 처리하기 위해서는 wp_ajax_nopriv로 시작하는 액션 혹은 사용해야 하는데 이것을 통해 비 로그인 사용자가 특정 Ajax 요청을 수행할 수 있기 때문에 적절한 보안 검사를 수행하지 않으면 이 과정에서 보안 취약점이 발생할 수 있다. [그림 6]에서 (a)는 인증되지 않은 사용자의 요청을 허용하는 wp_ajax_nopriv_ 기본 코드로 ChatGPT를 통해 (a)를 시큐어 코딩이 가능하도록 요청하였다. 그 결과(b)와 같이 입력값 검증 및 CSRF 보호를 추가한 코드를 제시해 WordPress.org 사이트에서 수정하고 Advanced XML Reader 0.3.4 플러그인에서 발생하는 XXE Injection 취약점 진단을 실행하였다. 실습환경의 수정된 워드프레스에서는 (a)의 \$_POST['data'] 값에 대해 아무런 검증 없이 처리되었고, 인증되지 않은 사용자의 요청으로 악의적인 웹사이트가 사용자의 세션을 이용해 요청을 위조할 수 있었다. 따라서 (b)와 같이 코드를 수정하고 실행한 결과 wp_verify_nonce() 함수에서 Ajax 요청에 포함된 nonce 값의 검증을 통해 이 값이 유효하지 않으면 요청을 거부하였다. 또한 sanitize_text_field() 함수로 사용자로부터 입력받은 데이터를 필터링하여, 악의적인 코드 삽입을 방지하였고, 데이터 출력 시 esc_html() 함수를 사용해 XSS 공격을 방지하는 등 Ajax 요청할 때 발생할 수 있는 보안 취약점이 발생하는 것을 확인하였다. 따라서 (b)의 코드로

수정해 실험한 결과 wp_verify_nonce() 함수로 Ajax 요청에 포함된 nonce 값 검증 시 사용자자는 Ajax 요청을 보낼 때 보안 nonce를 포함해야 하는데 이 값이 유효하지 않으면 요청이 거부되었다. 또한 사용자로부터 입력받은 데이터를 sanitize_text_field() 함수로 사용자로부터 입력 받은 데이터를 필터링해 악의적인 코드 삽입을 방지하였고 데이터를 출력할 때는 esc_html()을 사용해 XSS 공격을 방지하는 것을 확인하였다. 또한 Nginx 로그와 WordPress 자체의 활동 로그를 분석한 결과 WordPress 사이트에 대한 능동적 공격은 WordPress 자체의 핵심보다 선택적으로 설치된 플러그인에 초점을 맞추고 있다. 따라서 이 문제 해결을 위해 실험에서 ChatGPT를 통해 우리가 분석한 결과를 비교하였다. Nginx의 액세스 로그를 분석한 결과 [그림 7] (a)와 같은 패턴이 반복적으로 나타났고 WordPress의 보안 플러그인에서 제공하는 로그인 시도 기록을 분석한 결과 [그림 7] (b)와 같은 사용자 'admin' 계정에 대한 로그인 실패 시도가 다수 기록된 것을 확인할 수 있었다.

```
192.168.0.100 -- [01/Sep/2024:12:34:56 +0000] "POST /wp-login.php HTTP/1.1" 200 4523 "https://example.com/"
192.168.0.100 -- [01/Sep/2024:12:34:57 +0000] "POST /wp-login.php HTTP/1.1" 200 4523 "https://example.com/"
192.168.0.100 -- [01/Sep/2024:12:34:58 +0000] "POST /wp-login.php HTTP/1.1" 200 4523 "https://example.com/"

(a) Nginx Log Analysis

[01-Sep-2024 12:34:56] Login attempt failed for user 'admin' from IP 192.168.0.100
[01-Sep-2024 12:34:57] Login attempt failed for user 'admin' from IP 192.168.0.100
[01-Sep-2024 12:34:58] Login attempt failed for user 'admin' from IP 192.168.0.100

(b) WordPress Activity Log Analysis
```

Fig. 7. Active Attack Analysis on WordPress Sites

ChatGPT를 통해 WordPress 사이트에 대한 능동적 공격 분석을 비교하기 위해 “Nginx 로그와 WordPress 자체의 활동 로그를 분석해 줘”라고 했을 때 결과로 /var/log/nginx/access.log를 통해 POST 요청이 동일한 IP에서 발생하고 있는 것을 확인해 주었고 이를 해결하는 조치 방법으로 반복적인 POST 요청 IP의 차단 설정을 [그림 8]과 같이 제시해 주었다.

```
[nginx-wordpress-login]
enabled = true
filter = wordpress-login
action = iptables[name=WordPress, port=http, protocol=tcp]
logpath = /var/log/nginx/access.log
maxretry = 5
findtime = 600
bantime = 3600
```

Fig. 8. Fail2Ban settings for active attack defense

ChatGPT의 응답은 [그림 8]과 같이 /etc/fail2ban/jail.local 파일에 설정 방법을 자세히 안내했다. 설정 방법은 Fail2Ban을 사용하여 다수의 로그인 실패 시도를 10분(findtime = 600) 내에 5번(maxretry = 5) 이상 로그인 실패를 감지하고 공격자가 사용하는 IP 주소를 1시간(bantime = 3600) 동안 차단하였다. ChatGPT를 통해 얻은 방식으로 Nginx와 WordPress 로그를 분석하고 공격 패턴을 감지한 후 즉각적인 대응 조치를 취함으로써 WordPress 사이트에 대한 능동적 공격을 효과적으로 방어할 수 있었다.

IV. Conclusions

본 논문에서는 다양한 분야에서 활용되고 있는 ChatGPT를 이용해 시큐어 코딩의 활용 가능성을 연구하였다. 시큐어 코딩의 기본 요소인 SQL 인젝션 대응 방안 중 PreparedStatement 이외의 방법들도 존재하는지 ChatGPT에 질의하고, 이를 직접 적용해 문제를 해결할 수 있음을 확인하였다. 그러나 정적 분석 결과를 바탕으로 시큐어 코딩을 적용할 때 생성형 AI를 활용하기 위해 반드시 고려해야 할 두 가지가 있다. 첫째, 개발 환경과 소프트웨어 목적 등 구체적인 상황을 제시하여 해당 환경에 최적화된 대응 방안을 받는 것이다. 둘째, 단순히 다른 대응 방안이 있는지 추가 질문을 통해 다양한 결과를 제공받는 것이다. 따라서 ChatGPT에 구체적으로 질문하는 것이 중요하나, 다른 대안이 있는지 재차 질문하여 다양한 결과를 얻을 수 있다는 점도 반드시 고려해야 한다.

시큐어 코딩을 위해서는 ChatGPT에 구체적으로 질문하는 것이 중요하나, 다른 대안이 있는지 재차 질문하여 다양한 결과를 얻을 수 있다는 점도 반드시 고려해야 한다. 또한, 기업이나 기관에서 생성형 인공지능을 활용할 때는 정확성 검증, 최신 정보 반영, 타당성 확인 및 윤리적 고려를 염두에 두고 사용해야 한다. AI가 제안하는 보안 권장 사항이 모든 상황에 적합하지 않을 수 있어서, 실제 애플리케이션 및 시스템에 적합한지 확인하는 것이 필수적이다. 시큐어 코딩 작업에서 생성형 AI를 사용할 때는 윤리적 측면도 고려해야 하며, ChatGPT가 제시한 권장 사항이나 해결책이 오히려 보안 문제를 초래할 수도 있다는 점을 명심해야 한다. 소프트웨어 보안은 현대 사회에서 매우 중요한 요소로, 시큐어 코딩은 안전한 소프트웨어 개발뿐만 아니라 전반적인 안전을 위한 필수 작업이다. ChatGPT를 활용하여 많은 개발자는 생소한 보안 영역에 대한 정보를 쉽게 얻을 수 있으며, 적절한 대응 방안도 고려할 수 있을 것으로 기대된다.

REFERENCES

- [1] Statistics Korea, ‘Population Projections for Nationals and Foreigners Reflecting the 2021 Future Population Projections: 2020~2040’, 2022. 04. <https://eiec.kdi.re.kr/policy/materialView.do?num=225335>
- [2] S. B. Son, H. M. Lee, S. H. Park D. H. Kim and J. H Kim, “Trends in Autonomous Course of Action for Cyber Attacks Using Reinforcement Learning,” The Journal of Korea Information and Communications Society. Vol.11, pp. 1776-1787, November. 2022, DOI : 10.7840/kics.2022.47.11.1776
- [3] J. M. Suh and J. W. Kim, “Inducing Harmful Speech in Large Language Models through Korean Malicious Prompt Injection Attack,” The Journal of Korea Institute of Information Security & Cryptology, Vol. 34, No. 3, pp. 451-461, June, 2024, DOI : 10.13089/JKIISC.2024.34.3.451
- [4] Y. Tian, W. Cui, D. Deng, X. Yi, Y. Yang, H. Zhang and Y. We “ChartGPT: Leveraging LLMs to Generate Charts from Abstract Natural Language,” The Journal of IEEE Transactions on Visualization and Computer Graphics, Vol. , No. 1, pp. 1-15, February, 2024. DOI :10.1109/TVCG.2024.3368621
- [5] J. K. Lee, I. H. Ji, K. H. Jeon and J. T. Seo, “Proposal Of Ai-Based Anomaly Detection Model For V2X Communication Section,” in Proceeding of the Platform Technology Conferences, Vol. 10, No. 2, pp. 25-30. December, 2023. <https://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE11657687>
- [6] J. S. Choi and K. H. Kook, “Analysis on Domestic Secure Coding Rule using Secure Coding Rule Selection Evaluation Mode,” The Journal of Security Engineering, Vol. 11, No. 4, pp. 325-338, April, 2014, DOI :10.14257/jse.2014.08.02
- [7] Google Cloud, “The State of AI and Security Survey Report”, <https://www.itworld.co.kr/news/332707#csidx3e6f2480b23dc8589b4eb9ca78f2e01>
- [8] S. W. Lee(2021, December). Software Development Security Guide, Korea Internet& Security Agency[Online]. <https://www.kisa.or.kr/2060204/form?postSeq=5&page=2>
- [9] I. S. Jeon and K. S. Song. “Development of Block-based Code Generation and Recommendation Model Using Natural Language Processing Model,” The Journal of the Korean Association of Information Education, Vol. 26, No. 3, pp. 197-207, June, 2022. DOI : 10.14352/jkaie.2022.26.3.197
- [10] Y. J. Lee, J. M.Kang and J. H. Kim. “ Research on evasion attacks and LLM intrusion prevention models in GPT 4.0,” Journal of the Korea Academia-Industrial cooperation Society, Vol. 25, No. 3, pp. 10-15, March, 2024. DOI : 10.5762/KAIS.2024.25.3.10
- [11] L. Kree, R. Helmke, and E. Winter, “Using Semgrep OSS to Find OWASP Top 10 Weaknesses in PHP Applications: A Case Study. In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment,” Lecture Notes in Computer Science, Vol. 14828, pp. 64-83, July. 2024. DOI:10.1007/978-3-031-64171-8_4
- [12] D. Fucci, E. Alégroth, M. Felderer and C. Johannesson, “Evaluating software security maturity using OWASP SAMM: Different approaches and stakeholders perceptions,” The Journal of Systems and Software 214 [Online], Vol. 214, No. , pp.112062, August, 2024. DOI:10.1016/j.jss.2024.112062
- [13] E. Y. Yang. “Necessity of regulation on the development and use of generative AI -Focusing on Large language Models conversational A.I. services (LLMs AD)-,” The Journal of SungKyunKwan Law Review, Vol. 35, No. 2, pp. 293-325, June, 2023. DOI : 10.17008/skklr.2023.35.2.009
- [14] Y. T. Oh, and I. J. Jo. “Data Modeling for Cyber Security of IoT in Artificial Intelligence Technology,” The Journal of the Korea Contents Association, Vol.21, No.12, pp.57-65. December, 2021. DOI: /10.5392/JKCA.2021.21.12.057
- [15] K. S. Ko and I. J. Jo, “Application of Integrated Security Control of Artificial Intelligence Technology and Improvement of Cyber-Threat Response Process,” The Journal of the Korea Contents Association, Vol.21, No.10, pp.59-66. October, 2021. DOI : 10.5392/JKCA.2021.21.10.059
- [16] S. Hamer, M. d’Amorim and L. Williams, “Just another copy and paste? Comparing the security vulnerabilities of ChatGPT generated code and StackOverflow answers,” In 2024 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, pp. 87-94. May, 2024.
- [17] M. Jamdade and Y. Liu, “A Pilot Study on Secure Code Generation with ChatGPT for Web Applications,” In Proceedings of the 2024 ACM Southeast Conference, pp. 229-234, April, 2024. DOI:10.1145/3603287.365119
- [18] T. D. J. M. Sanguino, “Enhancing Security in Industrial Application Development: Case Study on Self-Generating Artificial Intelligence Tools,” The Journal of Applied Sciences, Vol.14, No. 9, pp.1-18, April, 2024. DOI:10.3390/app14093780

Authors



Young-Bok Cho received the M.S., and Ph.D. degrees in Computer Science from Chungbuk National University, Korea, in 2003 and 2012, respectively. also Dr. Cho received more Ph.D degrees in Medical and Law from Chungbuk

National University and Chungnam National University , Korea, in 2019 and 2024, respectively. She has Professor of Information Security at Daejeon University, Daejeon, Korea , in 2018 to 2024, She is currently a Professor in the Computer Education at Andong National University, Andong, Korea, in 2024. Her research interests include AI medical image processing, information security and medical information protection, mobile security.